REMARKS

This Amendment is submitted in response to the outstanding Office Action dated November 23, 2007. Claim 6 is amended hereby. Claims 1, 4-7, 11, 12, 15, 17-19 and 21 remain pending hereinafter, where claims 1, 15 and 21 are the independent claims.

At paragraph 8 of the Office Action, claim 6 was objected to. Applicants have amended claim 6 to change its dependency from 3 to 1.

At paragraphs 9 and 10 of the outstanding Office Action, claims 1-4, 15 and 17-19 were rejected under 35 USC §102(e) as anticipated by US Patent No. 6,721,941 to Morshed, et al. (Morshed). At paragraphs 11 and 12 of the Office Action, claims 21 and 11-12 were rejected under 35 USC §103(a) over Morshed in view of US Patent No. 6,839,893 to Bates, et al. (Bates). At paragraph 13 of the Office Action, claim 21 was further rejected under 35 USC §103(a) over Morshed in view of US Patent No. 6,161,196 to Tsai.

Response to rejections Under 35 USC §102(e)

With respect to independent claims 1 and 15, the Examiner asserts that Morshed discloses a software tool containing machine readable instructions stored on a physical medium for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules (Figs. 2-4 and related text), the software tool comprising:

a pattern detector manager including machine readable instructions for inserting into a running computer program a plurality of entry breakpoints (e.g., Fig. 14, blocks 442, 448, 452, 456, 460, 464, col. 23, line 1, to col. 24, line 11),

automatically, with little or no intervention from a use (e.g. col. 20, line 14-19 and 63-col. 21, line 50,

each of said entry breakpoints being associated with one of a plurality of defined coding patterns (e.g., Fig. 14, blocks 446, 450, 454, 458, 462); and

a plurality of pattern detectors, each of the pattern detectors being associated with one of said defined coding patterns, including machine readable instructions, and being invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding pattern associated with said each of the pattern detectors, is reached in the computer program (e.g., Fig. 15, col. 24, lines 11-62, Fig. 16, col. 25, lines 1-37),

for determining whether the computer program violates the coding pattern associated with said each of the pattern detectors (e.g., Fig. 12, col. 21, lines 6-67 and col. 23, line 36, through col. 24, line 11)

by inserting into the program at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with said one of the entry breakpoints (Block 442, 446, 448; col. 23, line 1-col. 24, line 11, col. 24, lines 47-57; Fig. 17; Fig. 14, 452, 456, 460, 464, 448);

wherein the plurality of defined coding patterns is selected from a group comprising best practice patterns and problematic coding patterns (col. 1, lines 24-36, col. 32, line 61-col. 33, line 27, col. 55, lines 31-47.

In reading the "Response To Arguments set forth at paragraphs 6 and 7 of the outstanding Office Action, and the Claim Rejections at paragraphs 9-13, applicants realize that Examiner Dao has not recognized the distinctions between the invention as claimed, and Morshed. Morshed utilizes a traditional debugger interface to gather execution information about a distributed application. Morshed does not teach or suggest a software tool as claimed; Morshed's debugger interface is used with instrumentation techniques, particularly for obtaining profiling information. Applicants' invention of claims 1 and 15 is directed to developing application programs, whereupon as the application is run by the debugger, the inventive tool operates to identify code patterns in the running application that violate a given set of coding rules.

Morshed does not disclose a software tool for monitoring the behavior of a running computer program to detect code patterns that violate a set of coding rules. Morshed places instrumentation instructions into compiler generated transitional representations, which eventually realizes instrumentation functionality in the fully compiled and fully operational object code 46. Morshed's Fig. 2 shows a data flow diagram illustrating a compiler operating in conjunction with IR code instrumentation. Morshed's Fig. 3 shows a data flow diagram illustrating interaction between the various stages of the compiler and IR code instrumentation, and Fig. 4 highlights detailed software operation.

The text describing Fig. 4 states that the compiler generates transitional representations of the source code transitioning to the operation object code. These transitional representations are accessed by the compiler 42. Code instrumentation software 50 executes on processor 22 to access the transitional representations to add instrumentation instructions for functionality in the object code. When the object code is executed, the thus added instrumentation functionality facilitates debugging. The transitional representations 48 include various data elements that are created and/or accessed by the compiler. Morshed does not monitor for coding patterns, nor suggests the benefit of monitoring for coding patterns.

That is, Morshed does not teach a software tool for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules or best practices patterns at program runtime. As discussed in detail in the present application, in any large software deployment, subtle coding defects can cause problems in successful deployment of the software. Tracking down these defects may be extremely difficult in the production environment because of a number of factors. One factor is that, in many cases, the symptoms are usually not unique to a particular type of software defect. This makes correlating symptoms to specific defects nearly impossible. Another factor is that many symptoms may manifest themselves only during production level loads and production configurations. This means that the defects are often undetected in the testing and debugging of software. In addition, the reasons rendering a piece of code defective are often complex, and may span third-party libraries and frameworks.

The present invention effectively addresses these issues by observing the behavior of a running program within the context of a large number of defined coding patterns, and automatically flagging violations of the coding patterns when they occur. Applicants' software tool comprises a pattern detector manager and a plurality of individual pattern detectors. The pattern detector manager is provided for inserting into the running computer program a plurality of entry breakpoints, each of which is associated with one of a plurality of defined coding patterns.

Morshed's code instrumentation software 50 operates with the compiler 42 to access transitional representations 48 and add to the representations instrumentation instructions for debugging the final object code version. Morshed does not mention a code pattern detector, or detector monitor. Each of applicants' code pattern detectors is associated with one of the defined coding patterns, and each pattern detector is invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding patterns that, in turn, is associated with the pattern detector, is reached in the computer program. When invoked, the claimed pattern detector determines whether the computer program violates the coding pattern associated with the pattern detector.

The prior art, and in particular Morshed, does not disclose or suggest the use of plural pattern detectors in the manner described above. Morshed's Fig. 14 and related text describe instrumenting a method of a class, Fig. 12 and related text describe the method instrumenting byte code, Figs 15-17 and related text describe portions of Fig. 14, and its class operation in

greater detail. Morshed does not teach the use of a debugger to automatically detect code patterns that violate a given set of coding rules. An example of a coding rule is: an invocation of method A should never follow an invocation of method B in a calling sequence. Such rules are generally impossible to verify prior to program runtime, because a compiler may not be able to enumerate all possible control flow paths through a static analysis of the program code. It is also impractical to enforce such a coding rule as a programming discipline.

Independent claims 1 and 15 describe important features not shown in or suggested by the prior art, for example, the use of the pattern detector manager and the plurality of pattern detectors. The pattern detector manager is used for inserting into the running computer program a plurality of entry breakpoints, each of which is associated with one of a plurality of defined coding patterns. Also, as described in claims 1 and 15, each of the pattern detectors is associated with one of the defined coding patterns, and invoked by the pattern detector manager after one of the entry breakpoints associated with the coding patterns that, in turn, is associated with that pattern detector, is reached in the computer program. These claims positively describe the further feature that, when invoked, a pattern detector determines whether the computer program violates the coding pattern associated with the pattern detector.

Independent claim 1 recites a software tool containing machine readable instructions stored on a physical medium for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules. The software tool includes a pattern detector manager including machine readable instructions for inserting into a running computer

program a plurality of entry breakpoints, automatically, with little or no intervention for a user, each of said entry breakpoints being associated with one of a plurality of defined coding patterns and a plurality of pattern detectors, each of the pattern detectors being associated with one of said defined coding patterns, including machine readable instructions, and being invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding pattern associated with said each of the pattern detectors, is reached in the computer program, for determining whether the computer program violates the coding pattern associated with said each of the pattern detectors by inserting into the program at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with said one of the entry breakpoints. The plurality of defined coding patterns is selected from a group comprising best practice patterns and problematic coding patterns.

Independent claim 15 sets forth a program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for monitoring the behavior of a running computer program. The method includes using a pattern detector manager to insert into a running computer program a plurality of entry breakpoints, each of said entry breakpoints being associated with one of a plurality of defined coding patterns, and monitoring to detect the occurrences of the entry breakpoints in the computer program, and upon detection of one of the entry breakpoints in the computer program, further inserting into the program at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with said one of the entry breakpoints and using a plurality of pattern detectors for monitoring the computer program, wherein each of the pattern detectors

are associated with one of said defined coding patterns, including the step of the program detector manager invoking each of the pattern detectors, after one of the entry breakpoints associated with the coding pattern associated with said each of the pattern detectors, is reached in the computer program, for determining whether the computer program violates the coding pattern associated with said each of the pattern detectors.

Morshed, at the figures and related text cited to reject independent claims 1 and 15, does not include the novel debugging operational use of two levels of breakpoint insertions by a pattern detection monitor associated with a set of coding rules, to detect code patterns that violate same coding rules. Morshed does not teach or suggest inserting entry breakpoints into a program, and during monitoring operation in which those entry breakpoints are detected, inserting at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with the entry breakpoints, the coding patterns selected from a group consisting of best practice patterns and problematic coding problems. Morshed does not seek to identify patterns based on rules.

While the Examiner indicates that Morshed's Fig. 14 flowchart is representative of applicants' claimed pattern detector manager, and that Fig. 14 shows each of the entry breakpoints associated with one of a plurality of defined coding patters, applicants respectfully disagree. Morshed's Fig. 14 discloses Morshed's method of instrumenting a class, and is not concerned with pattern instructions. Neither Fig. 14 nor the text describing Fig. 14 suggest inserting into the program at least one further breakpoint for identifying a respective step in the

program that is part of the coding pattern associated with one of the entry breakpoints. Morshed does not teach or suggest that the plurality of defined coding patterns are selected from a group comprising best practice patterns and problematic coding patterns, particularly not at col. 1, lines 24-36, col. 32, line 61-col. 33, line 27 and col. 55, lines 31-47. Morshed uses breakpoints, but does not insert further breakpoints, and does not look for improper coding patterns.

In view of the significant differences between Morshed and independent claims 1 and 15, applicants respectfully assert that claims 1 and 15 are not anticipated by Morshed under section 102. Claims 4-7 depend from claim 1 and are patentable therewith. Claims 17-19 depend from claim 15 and are patentable therewith. Applicants, therefore, respectfully request withdrawal of the rejections to claims 1, 4-7, 15 and 17-19 under Section 102 in view of Morshed.

Response to rejections Under 35 USC §103(a)

Morshed and Bates

At paragraphs 11 and 12 of the Office Action, claims 21 and 11-12 were rejected under 35 USC §103(a) over Morshed in view of US Patent No. 6,839,893 to Bates.

With respect to the rejection independent claim 21, the Examiner asserts that Morshed discloses a method of detecting code patterns in a computer program that violates a given set of coding rules, comprising the steps of:

defining a set of coding rules, each of the coding rules being associated with a respective one pattern detector (e.g., Fig. 14, blocks 446, 450, 454, 458, 462);

providing a pattern detector manager for managing said pattern detectors (e.g., Fig. 14, blocks 442, 448, 452, 456, 460, 464, col. 23, line 1, to col. 24, line 11);

providing a computer program, and running the computer program as a debug mode (e.g., Fig. 12, col. 21, lines 6-67 and col. 23, line 36, through col. 24, line 11);

the pattern detector manager identifying, during the running of the computer program in the debug mode, points in the computer program that relate to said coding rules (e.g., Figs. 15-17, col. 24, line 11, through col. 25, line 67), and said pattern detector manager inserting into the computer program an entry breakpoint at each of said identified points (col. 20-lines 40-49, and col. 20, line 63 through col. 21, line 5);

said pattern detector manager invoking each of the pattern detectors to monitor the computer program for a violation of the coding rule associated with said each of the pattern detector (col. 21, lines 6-67), including the step of, each of the pattern detectors inserting one or more further breakpoints into the computer program to identify further points in the computer program that relate to the coding rule associated with said each of the pattern detector (col. 24, line 11, through col. 25, line 67), and tracking said additional breakpoints to determine whether the computer program violates the coding rule associated with said each of the pattern detectors (e.g., Fig. 14, col. 23, line 1, through col. 24, line 11, col. 24, lines 47-57; col. 66, lines 20-31);

and wherein each of the pattern detectors monitors the computer program for the occurrence of any of the first set of defined conditions, the occurrence of which violates the coding rule associated with said each of the patter detectors (col. 1, lines 24-36; col. 32, line 61-col. 33, line 27; col. 55,line 31-47).

The Examiner further asserts that Morshed does not explicitly disclose monitoring the computer program for the non-occurrence of any one of a second set of defined conditions, the non-occurrence of which violates the coding rule associated with said each of the pattern detectors. The Examiner continues that Bates is analogous art, and discloses monitoring the computer program fro the non-occurrence of any one of a second set of defined conditions, the non-occurrence of which violates the coding rule associated with said each of the pattern detectors (Fig. 3, all of col. 31, col. 3, line 63-col. 4, line 21, Fig. 4b, col. 5, lines 29-67; Fig. 5; col. 6, line 52-col. 7, line 8). The Examiner concludes that it would have been obvious to combine Bates with Morshed in order to detect an unreachable breakpoint and display a warning.

Applicants respectfully disagree that Morshed and bates render obvious independent claim 21. Independent claim 21 sets forth a method of detecting code patterns in a computer program that violate a given set of coding rules. The method includes defining a set of coding rules, each coding rule associated with a respective one pattern detector of a set of pattern detectors, providing a pattern detector manager for managing said pattern detectors, providing a computer program, and running the computer program in a debug mode; the pattern detector manager identifying, during the running of the computer program in the debug mode, points in the computer program that relate to said coding rules. The pattern detector manager inserts into the computer program an entry breakpoint at each of said identified points and invokes each of the pattern detectors to monitor the computer program for a violation of the coding rule associated with said each of the pattern detectors.

The claim 21 method further includes that each of the pattern detectors inserts one or more further breakpoints into the computer program to identify further points in the computer program that relate to the coding rule associated with said each of the pattern detectors and tracking said additional breakpoints to determine whether the computer program violates the coding rule associated with said each of the pattern detectors. Each of said additional breakpoints identifies a respective step in the computer program that is part of the coding pattern associated with said one of the entry breakpoints, and wherein each of the pattern detectors monitors the computer program for the occurrence of any one of the first set of defined conditions, the occurrence of which violates the coding rule associated with said each of the pattern detectors and monitors the computer program for the non-occurrence of any one of a second set of defined conditions, the non-occurrence of which violates the coding rule associated with said each of the pattern detectors.

Morshed as cited by the Examiner does not include the novel debugging operational use of two levels of breakpoint insertions by a pattern detection monitor associated with a set of coding rules, to detect code patterns that violate same coding rules. Morshed does not teach or suggest a system that inserts into the program a number of entry breakpoints, and during monitoring operation in which those entry breakpoints are detected, inserting at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with the entry breakpoints. Morshed does not disclose selecting a plurality of defined coding patterns from a group consisting of best practice patterns and problematic coding problems. Morshed does not seek to identify patterns based on rules.

While the Examiner indicates that Morshed's Fig. 14 flowchart is representative of applicants' claimed pattern detector manager, and that Fig. 14 shows each of the entry breakpoints associated with one of a plurality of defined coding patterns, applicants respectfully disagree. Morshed's Fig. 14 discloses Morshed's method of instrumenting a class, and is not concerned with coding pattern instructions. Nowhere in Morshed's Fig. 14, or the text describing Fig. 14, does Morshed suggest inserting into the program at least one further breakpoint for identifying a respective step (in the program) that is part of the coding pattern associated with one of the entry breakpoints. Nowhere does Morshed teach or suggest that the plurality of defined coding patterns are selected from a group comprising best practice patterns and problematic coding patterns. Morshed uses breakpoints, but does not insert further breakpoints, and does not look for improper coding patterns.

Bates does not remedy the shortcomings of Morshed. Bates' Fig. 3 unreachable statement list is not the equivalent to applicants' claimed use of coding patterns, or monitoring the computer program for the non-occurrence of any of a set of defined conditions, the non-occurrence of which violates the coding rule associated with each pattern detector. With all due respect, applicants do not see how Bates combined with Morshed can realize the coding pattern centric method of independent claims 21. A warning to a user suggesting an unreachable breakpoint is not equivalent to applicants' claimed use of coding patterns, or monitoring the computer program for the non-occurrence of any of a set of defined conditions, the non-occurrence which violates the coding rule associated with each pattern detector. Still less does

Bates teach or suggest inserting further or additional breakpoints to identify a respective step in the computer program that is part of the coding pattern.

Applicants respectfully assert, therefore, that independent claim 21 is patentable in view of Morshed and Bates under Section 103(a). Claims 11 and 12 depend from claim 21 and are patentable therewith. Applicants, therefore, respectfully request withdrawal of the rejections to claims 21, 11 and 12 under Section 103(a) in view of Morshed and Bates.

Morshed and Tsai

Claim 21 was further rejected under 35 USC §103(a) over Morshed in view of US Patent No. 6,161,196 to Tsai.

With respect to the rejection independent claim 21, the Examiner asserts that Morshed discloses a method of detecting code patterns in a computer program that violates a given set of coding rules. The Examiner asserts that Morshed teaches a method comprising the steps of:

defining a set of coding rules, each of the coding rules being associated with a respective one pattern detector (e.g., Fig. 14, blocks 446, 450, 454, 458, 462);

providing a pattern detector manager for managing said pattern detectors (e.g., Fig. 14, blocks 442, 448, 452, 456, 460, 464, col. 23, line 1, to col. 24, line 11);

providing a computer program, and running the computer program as a debug mode (e.g., Fig. 12, col. 21, lines 6-67 and col. 23, line 36, through col. 24, line 11);

the pattern detector manager identifying, during the running of the computer program in the debug mode, points in the computer program that relate to said coding rules (e.g., Figs. 15-17, col. 24, line 11, through col. 25, line 67), and said pattern detector manager inserting into the computer program an entry breakpoint at each of said identified points (col. 20-lines 40-49, and col. 20, line 63 through col. 21, line 5);

said pattern detector manager invoking each of the pattern detectors to monitor the computer program for a violation of the coding rule associated with said each of the pattern detector (col. 21, lines 6-67), including the step of,

each of the pattern detectors inserting one or more further breakpoints into the computer program to identify further points in the computer program that relate to the coding rule associated with said each of the pattern detector (col. 24, line 11, through col. 25, line 67), and

tracking said additional breakpoints to determine whether the computer program violates the coding rule associated with said each of the pattern detectors (e.g., Fig. 14, col. 23, line 1, through col. 24, line 11, col. 20, lines 6-62); and

wherein each of the pattern detectors monitors the computer program for the occurrence of any of the first set of defined conditions, the occurrence of which violates the coding rule associated with said each of the patter detectors (col. 1, lines 24-36; col. 32, line 61-col. 33, line 27; col. 55,line 31-47).

The Examiner asserts that Morshed does not explicitly disclose monitoring the computer program for the non-occurrence of any one of a second set of defined conditions, the non-occurrence of which violates the coding rule associated with said each of the pattern detectors.

The Examiner continues that Tsai is analogous art, and discloses monitoring the computer program for the non-occurrence of any one of a second set of defined conditions, the non-occurrence of which violates the coding rule associated with said each of the pattern detectors (col. 10, lines 58-67). The Examiner then concludes that it would have been obvious to combine Tsai with Morshed in order to declare faults after a maximum wait threshold is reached, as suggested by Tsai at col. 10, lines 58-67.

Applicants respectfully disagree. Applicants' independent claim 21 sets forth a method of detecting code patterns in a computer program that violate a given set of coding rules, as described above in response to the rejection of independent claim 21 over Morshed and Bates under section 103(a). As already mentioned, Morshed does not include the novel debugging operational use of two levels of breakpoint insertions by a pattern detection monitor associated with a set of coding rules to detect code patterns that violate same coding rules. That is, Morshed does not teach or suggest a system that inserts into the program a number of entry breakpoints, and during monitoring operation in which those entry breakpoints are detected, inserting at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with the entry breakpoints, and still less that the plurality of defined coding patterns are selected from a group consisting of best practice patterns and problematic coding problems.

Tsai does not remedy the shortcomings of Morshed. Tsai's col. 10, lines 58-67, states that if a fault occurs on backend2 affects the target program such that it is unable to reach the

voting breakpoint, the frontend wait until a maximum wait threshold is reached and then declares backend2 to be erroneous. Applicants do not claim a voting breakpoint, or the backend2. Applicants' claim 21 invention monitors the computer program for problematic coding patterns, and the non-occurrence of any of a set of defined conditions, the non-occurrence of which violates the coding rule associated with each pattern detector.

Nether Morshed, Tsai or the proposed combination of Morshed and Tsai teach or suggest the claim 21 method of detecting code patterns in a computer program that violate a given set of coding rules. Use of voting breakpoints is not equivalent to applicants' claimed use of coding patterns, or monitoring the computer program for the non-occurrence of any of a set of defined conditions, the non-occurrence of which violates the coding rule associated with each pattern detector. The proposed combination does not teach or suggest inserting further or additional breakpoints to identify a respective step in the computer program that is part of the coding pattern.

Applicants respectfully assert, therefore, that independent claim 21 is patentably distinguishable from Morshed and Tsai under Section 103(a), and request withdrawal of the rejection of claim 21 under Section 103(a) in view of Morshed and Tsai.

Applicants, therefore, respectfully request withdrawal of the rejection of claims 1, 4-7, 11, 12, 15, 17-19 and 21 in view of Morshed, Bates and Tsai, alone or in any proposed combination. If the Examiner believes that a telephone conference with applicants' attorneys

would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

John F. Vodopia
Registration No.: 36,299
Attorney for Applicants

Scully, Scott, Murphy & Presser, P.C.
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 742-4343

JFV:tb